VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

METODOLOGIE ŘÍZENÍ PROJEKTŮ ZA POMOCÍ AGILNÍCH PŘÍSTUPŮ

AGILE PROJECT MANAGEMENT METHODOLOGY

2011 Bc. Erik Šivic

Zadání

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 22. června 2011

.....

Poděkování

Děkuji vedoucímu mé diplomové práce panu Ing. Přemyslu Soldánovi, CSc. za manažerský přístup, cenné rady a za trpělivost k průběhu vývoje diplomové práce.

Abstrakt

Diplomová práce se zabývá analýzou, návrhem a implementací informačního systému jako on-line tutoriálu pro řízení projektů. Informační systém aplikuje nově navrženou metodologii agilního řízení projektu, jehož významnou vlastností je díky jeho modularitě naprostá nezávislost na této metodologii. Tento koncept byl zvolen s ohledem na očekávání budoucích požadavků na změny v aplikované metodologii či dalšího rozšiřování systému, kde je dokonce připraven aplikovat neomezené množství dalších metodologií. Informační systém je naimplementován jako webová aplikace pomocí technologie ASP.NET, C# a MSSQL.

Klíčová slova:

Informační systém, správa a řízení projektů, on-line tutoriál, projektový manažer, ASP.NET, C#, MSSQL, modularita, webová aplikace.

Abstract

This thesis deals with analysis, design and implementation of information system as an online tutorial for project management. Information system applies newly designed agile project management methodology, whose major feature is due to its modularity complete independence from this methodology. This concept was selected with the anticipation of future requirements for changes in the methodology applied and further expansion of the system, which is even ready to apply an unlimited number of other methodologies. The information system is implemented as a web application using ASP.NET, C # and MSSQL.

Keywords:

Information systems, project management, on-line tutorial, project manager, ASP.NET, C#, MSSQL, modularity, web application.

Seznam použitých symbolů a zkratek

IS – Informační systém

PM – Projektový manažer

RUP – Rational Unified Process, je iterativní proces vývoje softwaru vytvořený IBM

PMP – Projekt management plán

PMBOK – Zkratka Project Management Body of Knowledge, jde o metodiku řízení projektů

PMI – Project Management Institute, organizace starající se o metodiku projektového řízení

OGC – Office of Government Commerce, organizace starající se o metodiku projektového řízení PRINCE 2

IPMA – International Project Management Association, organizace starající se o metodiky projektového řízení

RBS – Resource Breakdown Structure, rozkládá zdroje podle jejich typů a kategorií

WBS – Work Breakdown Structure, rozklad objemu práce na dílčí úkony

DP – Decision Point, v češtině rozhodovací bod, jde o milník, ve kterém je třeba učinit rozhodnutí o pokračování projektu

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Problematika projektového řízení stále nabývá na své popularitě. Po vzoru velkých korporací si dnes už i střední a malé firmy začínají uvědomovat význam a přínos projektového řízení. Léta praxe naučila společnosti, že řešení projektů ad-hoc je nemyslitelné, a že je nutností postupy při jejich řešení nějak popsat a standardizovat. Začaly tedy vznikat různé směrnice, kterých by se budoucí projekty měly držet. Nabíráním zkušeností z řešených projektů se tyto směrnice dále upravovaly a rozšiřovaly, až vznikly robustní ověřené metodologie pro řízení projektů.

Projektové řízení je dnes velmi komplexní záležitost a projektové manažery dělá kvalitními množství nasbíraných zkušeností. Řízení velkých projektů bývá velmi složité a vyžaduje opravdu velkou míru pozornosti. Pro tyto případy se začal vyvíjet software, který by celému procesu řízení projektů napomáhal.

Vlastní zájem o tuto problematiku byl nakonec motivací ke vzniku této diplomové práce, jejímž předmětem je vytvoření informačního systému pro správu a řízení projektů. Informační systém je v tomto případě zamýšlen jako online tutoriál, který povede uživatele krok za krokem celým životním cyklem projektu od jeho inicializace až po ukončení.

Jednotlivé kroky tutoriálu se zakládají na nově navržené metodice řízení projektů, jejíž návrh byl předmětem souvisejících diplomových prací Jakuba Štolfy (1) a Ondřeje Koběrského (2), které mi posloužily jako zadání pro specifikaci požadavků na vyvíjený systém. V textu této práce často používám různě skloňovaný výraz „navržená metodologie“, čímž referuji právě tuto nově navrženou metodologii od zmíněných kolegů.

Na počátku diplomové práce vytýčím základní teoretická východiska potřebná k pochopení řešené problematiky. Pokusím se zde objasnit základní pojmy a problémy z oblasti projektového řízení. Dále popíši nástroje a programovací jazyky použité pro návrh a implementaci systému.

Následující kapitolu věnuji seznámení s nově navrženou metodologií a prezentuji celý tento návrh formou zadání pro další vývoj software.

Cílem mé práce je z obdrženého návrhu metodologie, přiložených případů užití a spoluprací s jejich tvůrci (Jakub Štolfa a Ondřej Koběrský), zřetelně specifikovat požadavky na vyvíjený IS. Následně tyto požadavky zanalyzovat a utvořit tak abstrakci popisované problematiky reálného světa. Na základě této abstrakce vyhotovím návrh implementace IS, a to s co největším ohledem na možný budoucí rozvoj a škálovatelnost IS. Nakonec tento návrh implementuji.

2 Teoretická východiska

Kapitola zachycuje základní teoretická východiska, která byla nezbytná pro vypracování diplomové práce. Hlavní doménou byl návrh implementace a samotná implementace informačního systému. V teoretické části si nejprve uvedeme základní principy a pojmy z této oblasti. Jelikož se jedná o specifický typ IS pro řízení projektů, je vhodné si pro pochopení smyslu celé práce nejdříve osvojit základní termíny projektového řízení.

2.1 Teorie z oblasti řízení projektů

2.1.1 Projekt

Pro tento pojem existuje celá řada více či méně se odlišujících definic. Společným jmenovatelem všech je chápání projektu jako jednorázové akce, která se skládá z prostorově a časově ohraničeného souboru souvisejících činností, jejichž uskutečnění vede k dosažení určitého cíle. Jednoznačně určený cíl je základní charakteristickou vlastností projektu. Po jeho dosažení, případně po uplynutí času, projekt končí. Časová omezenost je další charakteristickou vlastností – každý projekt musí mít přesně stanovený začátek a konec. Realizace projektu probíhá zpravidla postupně po menších krocích stanovováním si a plněním dílčích cílů. V průběhu realizace jsou vyžadovány různé zdroje, ať už materiální nebo lidské. Těch bývá obecně omezené množství. Koordinace a správné nakládání s nimi je pro řízení projektu klíčové.

Realizace projektu je cílená na uspokojení zákazníka (označován jako vlastník projektu). Vlastník určuje hlavní směr projektu a má vždy rozhodující slovo, a to především díky tomu, že na projekt vyčleňuje finanční zdroje. Vývoj projektu je po celou dobu ovlivňován vnitřními či vnějšími faktory, což do něj zanáší jistou neurčitost přinášející určitá rizika. Úkolem projektového manažera je tyto rizika odhadnout a pokusit se je maximálně eliminovat.

2.1.2 Proces

Projekty se skládají z procesů. Podle (3) je proces po částech uspořádaná množina činností, které tvoří opakovatelným způsobem požadované výstupy na základě jednoho či více vstupů. Procesy tvořící projekt (projektové procesy) jsou vykonávány lidmi a dělíme je na dvě základní skupiny:

- procesy řízení projektů (popisují, organizují a vykonávají práci na projektu)
- produktově orientované (specifikují a vytvářejí produkt projektu)

Procesy řízení projektů mohou být dle (4) rozděleny do pěti skupin na procesy:

- inicializační, vedoucí ke vzniku či zahájení procesu (fáze projektu)
- plánovací, definující a upřesňující cíle a vybírající nejlepší způsob dosažení cílů
- realizační, koordinující lidské a další zdroje při uskutečňování plánu

- kontrolní, zajišťující dosahování cílů sledováním a měřením odchylek od plánu, aby mohla být popř. provedena nápravná opatření
- závěrečné, formující převzetí projektu

Produktově orientované procesy jsou typicky definovány životním cyklem projektu a mění se podle oblasti aplikace.

Projekt má definován svůj začátek a konec a v rámci svého životního cyklu prochází různými fázemi, jejichž počet se může lišit podle podrobnosti členění. Základní je členění projektu na čtyři fáze:

- koncepční (tvorba koncepce)
- návrh
- realizace
- předání

V jednotlivých fázích je kladen různý důraz na jednotlivé nástroje a techniky řízení projektů.

2.1.3 Projektové řízení vs. řízení projektů

Tyto pojmy bývají často nesprávně považovány za ekvivalentní, avšak jejich význam se ve skutečnosti výrazně liší. **Projektové řízení** (angl. project management) je manažerská disciplína zabývající se řízením firem formou projektů. Jde o hlavní manažerskou strategii projektově orientovaných firem. Naproti tomu **řízení projektů** (angl. management of projects) správně označuje soubor nástrojů a technik určených pro řízení jednotlivých projektů.

Projektové řízení

Jak již bylo výše uvedeno, jedná se o strategickou variantu organizační struktury společnosti a koordinaci projektů na základě dostupných lidských zdrojů. Firmy uplatňující tuto strategii jsou označovány za **projektově orientované.** Realizace složitých procesů zde probíhá použitím dočasné struktury organizace, zacílené na realizaci projektu. Říkáme, že se jedná o **organizaci podle projektu**. I zde však musí existovat specifická stálá organizační složka, zajišťující integrační funkce. Takové firmy jsou charakteristické svou strategií, organizační strukturou a kulturou. „Projektové řízení je bouřlivě se vyvíjející oblast managementu a poskytuje příležitost ke stálému zdokonalování.“ (4)

Řízení projektů

„Řízení projektu je uplatnění veškerých poznatků, dovedností, nástrojů a technik na aktivity (činnosti) projektu takovým způsobem, aby byly splněny požadavky na projekt“ (5 str. 8). Tyto techniky jsou podrobně rozpracovány a využívány již dlouhou dobu v praxi. Smyslem kvalitního řízení projektů je správné koordinování a využívání všech zdrojů a zajištění tak co nejefektivnějšího plnění dílčích úkonů vedoucích k úspěšnému dotažení projektu, splnění stanoveného cíle a uspokojení zadavatele projektu.

Pro úspěšnou realizaci projektu je důležité zvolit vhodný přístup k jeho řízení. Existují proto i obecné souhrny znalostí z této oblasti, jejichž cílem je nalézt a popsat ty nejlepší

ověřené praktiky použitelné u většiny projektů. Těmto souhrnům se říká metodologie (nebo také metodiky) řízení projektů.

2.1.4 Metodologie řízení projektů

Řízení projektu by v dnešní době již nemělo být založeno pouze na zkušenostech jednotlivce, ale mělo dodržovat určitá pravidla - tj. řídit se již prověřenými metodikami a standardy. Tématu řízení projektů na mezinárodní úrovni se věnují různé profesní organizace nebo organizace, které tyto standardy vydávají. Ty nejvýznamnější v tomto oboru jsou: PMI, IPMA, OGC. Existuje také mnoho oborových a dílčích metodik pro řízení projektů. Obecně nejznámější a světově nejrozšířenější metodiky a standardy pro řízení projektů jsou:

- **PMBOK** (Project Management Body of Knowledge) od PMI (5)
- **PRINCE2** (PRojects IN Controlled Environments) od OGC (6)

Tyto metodiky a svým způsobem standardy obsahují vše potřebné k řízení projektů různého charakteru a různých velikostí. Rozhodnutí o tom, jakou metodu pro řízení projektů zvolit, je závislé především na třech základních faktorech:

- organizaci (druh, kultura, vyspělost, velikost, způsob řízení, atd.), kde projekt probíhá
- na specifikaci projektu (samotný předmět a cíle, finance, harmonogram, priority, kapacity, rizika, vazba na portfolio projektů, atd.)
- na projektovém manažerovi, který projekt řídí (jeho zkušenostech s konkrétní metodikou)

Řízení projektů podle **PMBOK** se dělí na 9 znalostních oblastí a 5 procesních skupin (fází). Jednotlivé procení skupiny podle metodiky PMBOK:

- inicializační procesní skupina – definuje a autorizuje projekt nebo jeho fázi, provádí identifikaci zainteresovaných skupin, výstupem je Charta projektu
- plánovací procesní skupina – hlavní procesní skupina, detailně definuje cíle a plánuje činnosti potřebné k jejich dosažení, výstupem je Projektový plán
- prováděcí procesní skupina – spojuje lidské zdroje k naplnění Projektového plánu.
- monitorovací a kontrolní procesní skupina – sleduje postup průběhu prací oproti plánu tak, aby mohla být přijata nápravná opatření
- uzavírací procesní skupina – provádí přijetí výstupu projektu a formálně ukončuje fázi nebo celý projekt

Každá ze znalostních oblastí PMBOK obsahuje procesy potřebné ke zdárnému dokončení projektu. Všechny procesy mají přesně definované vstupy, nástroje a techniky, výstupy. PMBOK rozeznává tyto znalostní oblasti:

- management integrace
- management rozsahu
- management času
- management nákladů
- management kvality
- management lidských zdrojů
- management komunikace

- management rizik v projektu
- management obstarávání

PRINCE2 vzniklo pod patronací britské vlády a v současné době je vyvíjen úřadem Office of Government Commerce (OGC). OGC provádí rozvoj portfolia Best Praktice, tzv. The Official Portfolio, které je vyvíjeno k podpoře britských státních organizací k rozvoji schopností ve čtyřech hlavních disciplínách. OGC's Best Practice prezentují praktické a efektivní postupy, které vzešly sloučením zkušeností z mnoha úspěšných globálních společností. OGC Portfolio zahrnuje:

- PRINCE2 – projektové řízení
- MSP (Managing Successful Programmes) – řízení programů
- M_o_R (Management of Risk) – řízení rizik
- ITIL (Information Technology Infrastructure Library) – řízení IT služeb

PRINCE2 je velmi praktický přístup k řízení projektů různých rozsahů a v různých oborech a byl též poprvé vydán v roce 1996. Byl několikrát aktualizován v letech 1998, 2002 a 2007 do jeho konečné podoby 4. edice. Při správném nasazení může přinést výsledky velmi rychle. Vychází z několika základních zásad a doporučenou sadou dokumentů, ze které je možné vybrat, co je pro daný projekt třeba.

2.1.5 Agilní metody řízení projektů

„Přestože úspěšnost IT projektů se v posledních letech stále zlepšuje, prostor pro vylepšení je stále velký. Mnohokrát se stalo, že projekt skončil později, než bylo plánováno, významně překročil budget nebo neodpovídal požadavkům zadavatele. Bohužel takto končí stále přibližně 60 % IT projektů. Podíváme-li se blíže na uvedené důvody neúspěchu, zjistíme, že klasické metody řízení softwarových projektů selhávají na přílišné složitosti systémů, nevhodně definovaných požadavcích a jejich častých změnách ze strany zadavatele, či nepochopení ze strany softwarové firmy. Mezi další uvedené důvody patří nejasně stanovené cíle, nízká podpora managementu, malá škálovatelnost, či nezkušenost projektových manažerů. Agilní metody se zaměřují právě na výše zmíněné aspekty a minimalizují jejich negativní vliv.“ (7)

V dokumentu „The Agile Manifesto“ (8) je definováno 12 základních principů, na nichž jsou založeny agilní metody řízení projektů.

1. Naše nejvyšší priorita je uspokojit zákazníka pomocí brzkých a průběžných dodávek software, který přinese konkrétní hodnotu.
2. Jsou vítány změny požadavků i v pozdějších fázích vývoje. Agilní procesy využívají změnu přinášející zákazníkovi konkurenční výhodu.
3. Časté dodávání fungujícího software v rozsahu od několika týdnů do několika měsíců s tím, že preferované jsou kratší časové úseky.
4. Obchodníci a vývojáři musí pracovat po celou dobu projektu společně každý den.
5. Vytvářet projekty s okruhem motivovaných jednotlivců. Zabezpečit pro ně prostředí a podporu, kterou potřebují a věřit jim, že danou práci zvládnou.
6. Nejúčinnější a nejefektivnější metodou předávání informací v rámci vývojového týmu je osobní komunikace.
7. Fungující software je hlavním/nejdůležitějším měřítkem pokroku projektu.

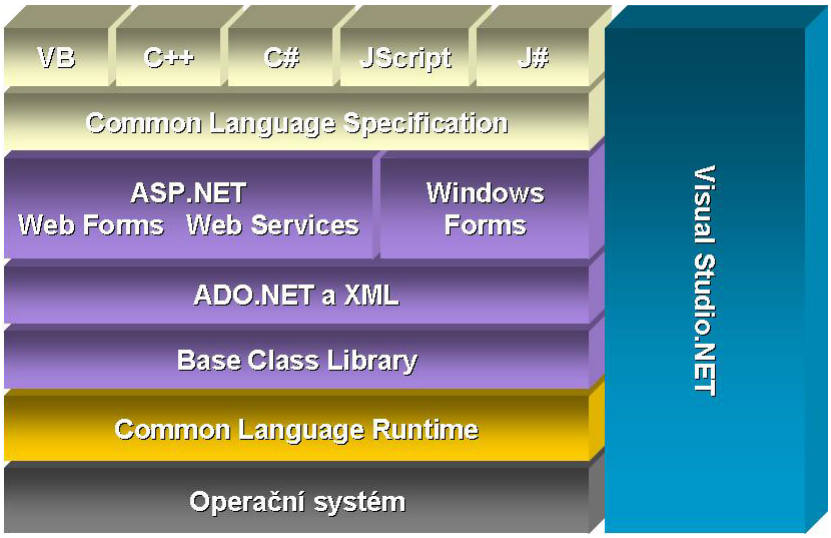
8. Agilní procesy podněcují udržitelné tempo vývoje. Sponzoři, vývojáři a uživatelé by měli být schopni toto tempo udržet po neomezenou dobu.
9. Neustálá pozornost směřovaná na technickou dokonalost a dobrý návrh zvyšuje agilitu (svižnost, hbitost).
10. Jednoduchost – umění maximalizovat množství práce, která se nemusí udělat je zásadní.
11. Týmy se samostatnou vnitřní organizací (samostatně organizující se týmy) vynikají nejlepšími návrhy, architekturou i požadavky.
12. V pravidelných intervalech by tým měl uvažovat nad tím, jak může být více efektivní a přizpůsobit tomu chování.

2.2 Programovací jazyky a nástroje pro implementaci

V této kapitole uvedu přehled základních technologií, kterých jsem využíval při realizaci diplomové práce.

2.2.1 ASP.NET

ASP.NET je platforma postavená na frameworku .NET pro programování webových aplikací. Architektura tohoto frameworku je zachycena na obrázku Obrázek 2-1 **Architektura platformy .NET** (9)Obrázek 2-1. Programování pod ASP.NET se zakládá na tzv. Common Language Runtime (CLR), což programátorům přináší možnost psát pro ASP.NET ve všech programovacích jazycích, které CLR podporuje. Nejčastěji využívané jsou Visual Basic .NET nebo C#. Webové aplikace v ASP.NET se oproti ASP, nebo i PHP, nemusí interpretovat, jsou předkompilované a jejich výhodou je i vyšší rychlost. ASP.NET je také objektově orientované a pro vývoj aplikací na této platformě existuje silné vývojové prostředí Visual Studio od společnosti Microsoft, v aktuální verzi Visual Studio 2010.



Obrázek 2-1 Architektura platformy .NET **(9)**

WebForms

Koncept ASP.NET WebForms ulehčuje programátorům přechod od programování klasických aplikací pro Windows do webového prostředí. Stránky se skládají z objektů - ovládacích prvků (Controls). Ty jsou protějškem ovládacích prvků ve Windows. Při tvorbě webových stránek je tedy možné používat ovládací prvky jako např. tlačítko (Button), nápis (Label) nebo složitější prvky jako např. GridView (zobrazuje data ze zdroje v tabulce). Existuje také možnost si vytvořit ovládací prvky vlastní, uložit je do knihovny a mít tak možnost je kdykoliv použít. Existují tři typy vlastních ovládacích prvků:

- Custom Controls – rozšiřují základní třídu Control
- Composite Controls – složené ovládací prvky
- User Controls – možnost navrhovat prvek jako klasickou aspx stránku

Ovládacím prvkům lze přiřazovat určité vlastnosti, zachytávat na nich události, atd. Webové ovládací prvky produkují HTML kód tvořící část výsledné stránky poslané do klientova prohlížeče.

Uchovávaní stavů

Webový protokol HTTP je sám o sobě bezstavový, proto se vyžaduje zachování stavu jinou metodikou. ASP.NET řeší tento problém kombinací HTML a JavaScriptu pomocí dvou základních technik.

ViewState uchovává informace mezi událostmi, tzv. postbacky (= opakované odesílání formuláře na server), v rámci jedné stránky v zakódovaném tvaru. Jde o obdobu skrytých formulářových polí. Využívá přitom pouze HTML a nevyžaduje žádnou zvláštní podporu na straně serveru ani klienta. Nevýhodou je, že mezi serverem a klientem dochází k přenosu většího objemu dat, a to zejména v případě, je-li ViewState využíván nesprávně.

SessionState naopak od ViewState ukládá veškeré informace na straně serveru a předává (typicky jako **cookie** nebo součást URL) pouze jednoznačný identifikátor. Toto řešení sice snižuje objem přenášených dat, ale na druhou stranu klade vyšší nároky na výkon serveru. Nejsou-li session používány správně, může se server stát náchylným k Denial of Service útokům. Session lze uložit do databáze, což přináší zjednodušení jejich použití ve webových farmách, zvyšuje výkon a navíc umožňuje zachovat stav i při restartu serveru.

2.2.2 MS SQL

Jedná se o systém řízení báze dat od společnosti Microsoft. Databázové servery společnosti Microsoft jsou obecně velmi oblíbené, což dokazuje i jejich rozšíření. V oblasti databázových technologií se řadí mezi naprostou špičku. Poskytují řadu technologických nástrojů pro potřeby datové logiky a bezpečnosti a také obsahují prostředky usnadňující spolupráci s Visual Studiem a .NET Framework. Díky této podpoře je server oblíben při vývoji aplikaci v .NET Frameworku. Nabízí také řadu nástrojů a funkcí pro zajištění zabezpečení.

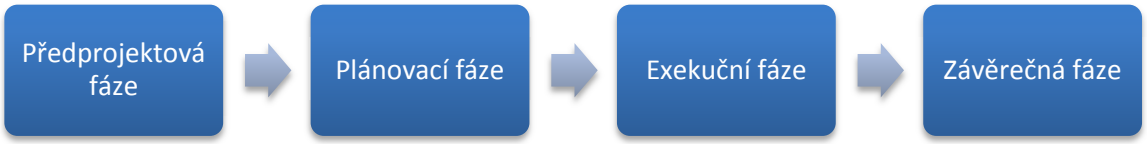
3 Specifikace požadavků

Na počátku je třeba uvést základní požadavky kladené na výsledný produkt. Ty vychází především z návrhu nové metodologie řízení projektů, viz (1) a (2). Uvedu tedy jen základní výtah podstatný pro cíl této práce. Z výsledné metodologie pak specifikuji požadavky na vyvíjený IS, provedu analýzu dat a funkcí, které budou systémem zpracovávány a poté udělám návrh jeho implementace.

3.1 Popis navržené metodologie

Stručně popíši navrženou metodologií řízení projektů, která byla předmětem diplomových prací (1) a (2). Uvádí se zde čtyřetapový životní cyklus projektu (Obrázek Obrázek 3-1) skládající se z:

- **Předprojektová fáze** – analyzují se možnosti projektu a rozhoduje se, zda vůbec projekt provádět
- **Plánovací fáze** – zabývá se naplánováním projektu, personálním sestavením projektu a dalšími aspekty
- **Exekuční fáze** – dochází k samotnému provádění projektu. Vychází se z agilních principů řízení projektů
- **Závěrečná fáze** – věnuje se uzavření projektu a uchováním informací projektu pro pozdější využití



Obrázek 3-1 Fáze navržené metodologie

Metodologie je navíc specifická tím, že dává dohromady tradiční přístupy k řízení projektů a agilní přístupy. Agilně se chová především v Exekuční fázi.

3.1.1 Předprojektová fáze

Všechny procesy v předprojektové fázi spějí k hlavnímu rozhodnutí, zda projekt přijmout či nikoliv. Projekt samotný vzniká nápadem vlastníka projektu nebo jeho sponzora. Vlastník projektu poté provede analýzy projektu, jeho cílů, předpokladů a rizik. Na základě těchto analýz pak rozhodne, jestli je projekt přínosný pro provádějící organizaci. Samotné rozhodnutí o přijetí a nepřijetí projektu se provádí na základě výstupů těchto procesů:

1. Vytvoření Předběžného rozsahu projektu - tento dokument shrnuje základní informace o projektu, které budou postupem času více upřesňovány.
2. Vytvoření a ohodnocení LFM (Logical Frame Matrix v překladu Matice logického rámce), LFM je analýzou zaměřená na detaily projektu, jeho aktivity, předpoklady a možná rizika.
3. Vytvoření finančních analýz typu ROI (Return of Investment v překladu Návratnost investic), NVP (Next Present Value v překladu Čistá současná hodnota) a doby návratnosti investic.
4. Identifikace a řízení rizik, která mohou ovlivnit projekt a plánování reakcí na možná rizika.

Podrobné zpracování Předprojektové fáze provedl ve své diplomové práci Jakub Štolfa (1)

3.1.2 Plánovací fáze

Plánovací fáze v rámci naší metodiky vychází hlavně z přístupů PMBOK. Slouží pro naplánování projektu tak, aby splnil všechny jeho stanovené cíle, omezení a uspokojil všechny zúčastněné strany. Plánovací fáze se skládá z několika procesů, které projektového manažera postupně provedou celou fází (například proces pro naplánování rozsahu, atd.).

Pokud projektový manažer nebyl zvolen v Předprojektové fázi, je zvolen na začátku Plánovací fáze. Projektový manažer si poté případně vybere svůj projektový tým, který mu pomáhá s průběhem Plánovací fáze. Později obsahem Plánovací fáze je i příprava personálního plánu, kde jsou určeny počty lidí pro Exekuční fázi projektu. Tito lidé potom v Exekuční fázi projektu tvoří vývojový tým.

Průběh Plánovací fáze spočívá v postupném vytvoření Projekt management plánu pro řízení rozsahu, času, nákladů, lidských zdrojů, kvality a komunikace. V rámci vytváření daného plánu jsou důležité dva nástroje:

1. WBS editor – umožňuje detailní rozložení rozsahu projektu na jednotlivé pracovní aktivity
2. Gantt diagram editor – vytváří časový harmonogram projektu z aktivit vytvořených ve WBS editoru. Aktivitám je přiděleno potřebné množství zdrojů.

Výhodou metodiky je, že jsou dané nástroje propojeny a naplánování zdrojů probíhá pro celou organizaci.

Podrobné zpracování Plánovací fáze provedl ve své diplomové práci Ondřej Koběrský (2).

3.1.3 Exekuční fáze

Exekuční fáze je provedena pomocí agilní metodiky SCRUM, která projekt rozděluje na krátká časová období nazvaná sprinty. Sprinty jsou řízeny Scrum masterem, jeden sprint běžně trvá jeden, nebo dva týdny. Scrum master určuje práci pro jednotlivé sprinty tak, že z vysledované výkonnosti týmu určuje reálný objem úkolů pro následující sprinty. Projektový tým tak ve sprintech postupně vytváří produkt projektu.

Podrobné zpracování Exekuční fáze provedl ve své diplomové práci Jakub Štolfa (1).

3.1.4 Závěrečná fáze

Závěrečná fáze navržené metodiky zahrnuje formální ukončení a odsouhlasení výstupů daného projektu. Výstupem ze Závěrečné fáze je dokument Závěrečná zpráva. Tento dokument obsahuje metriky projektu (plánované náklady/skutečné náklady, plánovaná doba/skutečná doba), a dokumentuje nabyté zkušenosti a doporučení z řízení projektů v několika určených kategoriích. Závěrečná zpráva poté slouží pro projektového manažera, respektive manažery, dalších projektů, kteří podle základních metrik svého projektu mohou najít dokument Závěrečné zprávy s podobnými metrikami. Díky tomu mohou tak načerpat zkušenosti a nápady pro zlepšení.

Podrobné zpracování Závěrečné fáze provedl ve své diplomové práci Ondřej Koběrský (2).

3.2 Požadavky na vyvíjený IS

Tato část se věnuje specifikaci základních funkčních i nefunkčních požadavků kladených na vyvíjený IS. Obecně je proces získávání konkrétní specifikace požadavků založen na komunikaci se zadavatelem projektu. Jde o to získat od něj co nejkvalitnější informace, protože zde se poprvé podrobně seznamujeme s jeho vizí o výsledném produktu. Je velmi důležité, udělat si jasnou představu o vyvíjeném IS, a to především představu shodnou s představou zadavatele. Chyby vzniklé v této fázi projektu díky neporozumění požadavkům zadavatele se mohou ve finále projevit jako fatální a přivést celý projekt do záhuby. Cena takových chyb bývá, není-li včas odhalena, velice vysoká. Na druhou stranu také u zadavatele se očekává aktivní zapojení a ochota spolupracovat. Jelikož se často může jednat o projekt týkající se nám naprosto vzdáleného oboru, je dobré si vytvořit společný slovník a jednoznačně si popsat některé důležité pojmy.

Výše zmíněný proces jsem při své práci musel aplikovat i já, a to většinou formou osobních schůzek s autory navrhované metodologie a vedoucím diplomové práce. I my jsme zde museli vyvinout společné úsilí pro stanovení si konkrétní a jasné představy o vyvíjeném produktu. Zejména výstupy těchto schůzek byly hlavním přínosem získávání specifikace požadavků na budoucí IS. Finálním výstupem všech schůzek jsou dva základní průběžně zpracovávané dokumenty:

- dokument popisující základní vizi budoucího informačního systému, tzv. Dokument vize (angl. Vision document**)**
- dokument popisující architekturu vyvíjeného software (tzv. SAD, z angl. Software Architecture Document)

Tyto dokumenty pochází z RUP. Popisují společnou představu o finálním produktu jak z pohledu funkčních, tak i nefunkčních požadavků a základní popis požadované architektury software.

Součástí navržené metodologie pak jsou i případy užití popisující možné scénáře interakce uživatele se systémem při procházení jednotlivých fází projektů a vykonávání konkrétních procesů v nich. Všechny diagramy se nacházejí na CD v příloze této práce.

K vytvoření ucelené specifikace požadovaných vlastností a chování vyvíjeného IS jsem měl k dispozici následující podklady:

- dokument vize
- dokument architektury software
- popis navržené metodologie
- realizace případů užití vč. diagramů
- vlastní zápisky ze schůzek
- a další písemně nezaznamenané poznatky získané při osobních schůzkách, či jiné vlastní nápady

Analyzováním výše zmíněných podkladů jsem vytvořil finální specifikaci požadavků využívající po vzoru metody BORM šablonu následujících otázek:

- PROČ?
- K ČEMU?
- KDO?
- VSTUPY?
- VÝSTUPY?
- OKOLÍ?

Odpovědi na položené otázky jsou souhrnem všech dosavadních vstupů a jsou uvedené v následující kapitole.

3.3 Výsledky specifikace požadavků

3.3.1 Proč nový IS?

Proč vůbec vyvíjet nový IS? Na trhu existuje celá řada IS pro správu a řízení projektů ať už komerčních, tak i zdarma dostupných. Většinou se však tyto systémy omezují pouze na určitý výsek z celého životního cyklu projektu nebo nemají požadovanou funkcionalitu či nesplňují jiné požadavky. Zásadní je ale fakt, že žádný existující IS neimplementuje nově navrženou metodologii. Navíc i původní záměr jejího návrhu počítá s realizací nového IS.

3.3.2 K čemu?

Potřebujeme tedy IS umožňující správu projektů a jejich řízení ve formě postupného procházení jednotlivých fází a procesů dle specifikace navržené metodologie.

3.3.3 Kdo?

IS bude sloužit především projektovým manažerům pro spravování a řízení projektů, tzn. plnění jednotlivých procesů a zakládání nových projektů. V Exekuční fázi projektu bude jako řídící osoba vystupovat SCRUM master. Do svých projektů může také nahlížet „zadavatel projektu“, ten však smí projekty pouze prohlížet, ale nemá možnost s nimi pracovat. Zvláštním typem uživatele je „člen hodnotící komise“, který vystupuje pouze v procesu Metoda logického rámce jako hodnotitel vyplněné matice. Dále bude v systému vystupovat Administrátor pracovního prostoru, který vznikne automaticky po založení

pracovního prostoru, mající na starosti správu vlastního pracovního prostoru a uživatelů v něm.

3.3.4 Vstupy

Účelem této části je nalezení všech atributů jednotlivých typů entit, které budou sloužit jako vstupy pro uživatele IS. Tyto atributy vystupují v konečném uživatelském rozhraní většinou jako textová pole formulářů, která uživatel sám manuálně vyplňuje.

- **Pracovní prostor**
  U pracovního prostoru budeme evidovat jednoznačné identifikační číslo, název pracovního prostoru a případně krátký popisek.

- **Typy zdrojů**
  Na pracovní prostor se vážou typy zdrojů využitelné pro projekty. U nich budeme zaznamenávat název, popis, celkové množství dostupných zdrojů, jejich zkušenosti a zvláštní vlastnosti a samozřejmě jednoznačné ID.

- **Projekt**
  Základním stavebním kamenem systému bude projekt. Ten bude určen opět svým jednoznačným ID a uchovává údaje o názvu projektu a jeho popis. Tato evidence je později rozšířena v procesu předběžný rozsah projektu.

- **Proces**
  Projekt se skládá z jednotlivých procesů, kde každý má svůj název, popis, označení, jedná-li se o rozhodovací bod či nikoli a do které fáze projektu je zařazen. Musí být určeno pevné pořadí těchto procesů a můžou existovat procesy vnořené.

- **Předběžný rozsah projektu**
  Předběžný rozsah projektu doplňuje základní evidenci projektů o informace o zadavateli/vlastníkovi projektu (jméno a organizace), cíle a přínos projektu, časové rozmezí projektu (odhadovaný začátek a odhadovaný konec) a odhadované finanční náklady.

- **LFM** (zkratka z angl. Logical Frame Matrix, česky nazýván Metoda logického rámce). Tento proces je složen z následujících podprocesů, které je nutno plnit sekvenčně v daném pořadí:

  - **Matice LFM -** tento proces je vyobrazen tabulkou, kde sloupce tvoří logické kroky/hierarchie cílů/intervenční logika, objektivně ověřitelné ukazatele, zdroje a prostředky ověření ukazatelů, předpoklady a rizika projektu. Pro každý sloupec tabulky budeme evidovat celkový cíl projektu, specifický cíl/účel projektu, očekávané výsledky a výstupy projektu, klíčové aktivity/činnosti, a předběžné podmínky a předpoklady.

  - **Ohodnocení LFM -** v tomto procesu evidujeme seznam hodnotících otázek (číslo otázky a samotná otázka) a jejich odpovědí/hodnocení.

- **Analýzy**
  Proces analýzy je také složen z několika podprocesů, jejich plnění však není sekvenční, nýbrž nabízí uživateli možnost volby, jakou analýzu si přeje provést.

  - **Jednoduchá analýza** eviduje pouze náklady, výnosy projektu a návratnost investice. Dodatečně evidujeme hodnotící kritéria pro určení výsledného doporučení, a sice všechny možné výsledky analýzy rozdělené do intervalů a body pro každý interval.

  - **Pokročilá analýza** eviduje časovou jednotku (roky nebo měsíce), diskontní sazbu, délku trvání projektu a náklady a výnosy pro každé časové období. Zde také dodatečně evidujeme hodnotící kritéria pro určení výsledného doporučení, stejně jako u jednoduché analýzy, zde však navíc zvlášť pro každou časovou jednotku.

- **Řízení rizik**
  Zde evidujeme číslo rizika, samotné riziko a kategorii, do které patří, pravděpodobnost jeho vzniku, míru jeho dopadu a protiopatření.

- **WBS** (akronym z angl. Work Breakdown Structure, česky Strom Rozkladu Práce)
  Při plnění tohoto procesu uživatel pracuje s několika následujícími typy entit. Nejedná se tedy o potomky procesu ani samostatné procesy.

  - **Komponenty WBS** (typ entity)
    Tyto komponenty tvoří strom rozkladu práce. Rozlišujeme dva druhy komponent:

    - **Pracovní balíček** - eviduje číslo účtu, název komponenty, popis a vlastníka. Dále celkový součet všech nákladů a celkovou délku trvání vypočtenou ze všech dalších komponent obsažených v tomto balíčku.

    - **Aktivita** - eviduje číslo účtu, název, popis, vlastníka a finanční náklady na tuto aktivitu a délku trvání.

  - **Milníky** (typ entity)
    Evidence milníků obsahuje datum, název milníku, jeho popis a označení, zdali se jedná o rozhodovací bod či nikoli.

- **Ganttův diagram**
  Proces vychází z WBS a doplňuje evidenci aktivit o typy použitých zdrojů a jejich množství.

- **Požadavek na změnu**
  Zde evidujeme číslo požadavku a samotný požadavek.

3.3.5 Výstupy ze systému a procesy

V této fázi datové analýzy se určují výstupní sestavy či procesy IS, které jsou zpracovány/vypočítány automaticky informačním systémem na základě zadaných vstupů od uživatele či dílčích výpočtů.

- Spojový seznam všech procesů podle specifikace navržené metodologie

- Aktuální proces
- Předešlý/následující proces
- Rodič procesu
- Potomci procesu
- Výpis uživatelů v pracovním prostoru
- Výpis projektů v pracovním prostoru
  - Aktuální projekty – výpis právě probíhajících projektů
  - Všechny projekty – výpis všech projektů
  - Mé projekty – výpis všech projektů, ke kterým je přiřazen konkrétní uživatel
- Výpis projektů v pracovním prostoru
- Založení nového projektu
- Výpis stavů všech procesů v konkrétním projektu
- Výpisy konkrétních procesů
  - Předběžný rozsah projektu
    - Výpis formuláře pro předběžný rozsah projektu daného projektu
  - Matice logického rámce (LFM)
    - Výpis tabulky LFM pro daný projekt
  - Ohodnocení LFM
    - Výpis všech hodnotících otázek
    - Výpis všech otázek včetně ohodnocení pro danou LFM
    - Celkové ohodnocení LFM pro daný projekt
    - Doporučení systému na základě hodnocení LFM pro daný projekt
  - Jednoduchá analýza
    - Výpis formuláře jednoduché analýzy pro daný projekt
    - Návratnost investice
    - Doporučení na základě vypočtené návratnosti investice
  - Pokročilá analýzy
    - Výpis formuláře pokročilé analýzy pro daný projekt
    - Čistá současná hodnota
    - Návratnost investice
    - Doba návratnosti investice
    - Doporučení na základě výsledků výpočtů analýzy
  - Řízení rizik
    - Výpis všech rizik v konkrétním pracovním prostoru
    - Výpis všech rizik v konkrétním projektu
    - Vyhledání rizik dle zadaných kritérií
  - WBS (Work Breakdown Structure)
    - Vykreslení stromu WBS pro daný projekt
    - Výpis všech komponent WBS pro daný projekt
    - Celková cena pracovního balíčku (součet cen ze všech komponent v podstromu tohoto balíčku)
    - Celková doba trvání pracovního balíčku (součet doby trvání všech komponent v podstromu tohoto balíčku)
    - Výpis dostupných typů zdrojů pro daný pracovní prostor
    - Výpis všech milníků pro daný WBS
    - Výpis konkrétního milníku pro daný WBS
  - Ganttův diagram

        - Výpis všech úkolů pro daný projekt v podobě Ganttova diagramu
    - Požadavek na změnu
        - Výpis všech požadavků na změnu pro daný projekt
        - Výpis konkrétního požadavku na změnu
- Registrace pracovního prostoru
- Registrace nového uživatele do pracovního prostoru

3.3.6 Okolí systému

Okolí systému budou vytvářet následující typy uživatelů:

- Uživatel - registruje se do systému
- Hodnotitel - provádí hodnocení matice LFM
- Hlavní uživatel - spravuje pracovní prostor a projekty
- Scrum master - řídící role v Exekuční fázi projektu
- Administrátor - správce celé aplikace

3.3.7 Nefunkční požadavky

Všechny nefunkční požadavky jsou detailně specifikovány v dokumentu Vize v příloze této práce. Zde si uvedeme jen jejich souhrn.

Lokalizace
Systém by měl podporovat změnu jazyků a měny.

Škálovatelnost
Systém může být rozšířen i na další klienty a není omezen počtem uživatelů, kteří mohou vstoupit do systému. Systém by měl být navržen tak, aby umožňoval snadné zásahy do použité metodologie či umožňoval přidání dalších metodologií.

Použitelnost
Vyžaduje se snadné intuitivního rozhraní systému a dostupnost. Systém musí být dostupný 24h denně, kromě doby zálohování.

Integrita dat
Data musí být ochráněna před pády systému.

Systémové požadavky
Produkční server - Windows Server 2003 a vyšší s databází podporující SQL příkazy

Webový server - Windows Server 2003 a vyšší

Klienti - klientům stačí běžné kancelářské PC s připojením k internetu a webovým prohlížečem.

Požadavky na prostředí
Produkční server musí být přístupný 24 hodin denně, je klíčový pro fungování systému. V případě výpadku serveru musí být co nejdříve obnoven.

Požadavky na zabezpečení
Práce se systémem je umožněna pouze přihlášeným uživatelům. Každý uživatel musí mít vytvořen účet a být přiřazen do pracovního prostoru. Uživatelé se přihlašují pomocí svého jména a hesla. Uživatelům jsou přiřazovány role, podle kterých se ověřuje přístupnost konkrétních funkcí systému.

Osobní údaje o uživatelích musí být chráněny. Heslo bude uloženo v zašifrované podobě pomocí algoritmu MD5.

Databáze se zálohuje jedenkrát denně v nočních hodinách.

4 Analýza a návrh

Vstupem této fáze je dokument specifikace požadavků jakožto výstup předchozího procesu. Zde jsou sepsané požadavky podrobeny analýze za účelem nalezení vhodné abstrakce reality. Analýza se provádí datová a funkční. Datová analýza slouží pro návrh datové struktury IS. Funkční analýza pak pro nalezení všech požadovaných funkcí systému.

Existují různé techniky a nástroje pro analýzu i návrh. Některé z nich slučují analýzu datovou a funkční v jeden úkon přinášející oba výsledky. V poslední době s rostoucí popularitou objektově orientovaného programování (OOP) se často mluví o objektově orientované analýze (OOA) či objektově orientovaném návrhu (OOD).

Objektově orientovaná analýza rozpoznává objekty mající své specifické chování a stav. Následně musí určit vzájemné vztahy mezi objekty a na základě společných vlastností je rozdělit do tříd.

Případy užití dodané s návrhem metodologie k této práci jsou právě konkrétním případem nástroje slučujícího datovou a funkční analýzu v jedno. Digramy případů užití (angl. Use Case), jsou součástí jazyka UML, který je přímo orientován na grafický popis objektově orientovaných systémů.

Jelikož dodané případy užití popisují jednotlivé procesy a fáze řízení projektu, zaměřím se zde především na datovou analýzu samotného funkčního jádra IS.

4.1 Datová analýza jádra systému a návrh databáze

Jak již bylo výše zmíněno, datová analýza si klade za cíl nalezení vhodných abstrakcí reálných subjektů. Zde jsem opět na základě specifikace požadavků identifikoval základní typy entit zastávajících funkční jádro systému (konceptuální úroveň):

- **Workspace** (pracovní prostor)
- **Projekt**
- **Proces** (obecný proces, neuchovává ani neodkazuje na atributy konkrétních procesů)
- **Uživatel**
- **Role**
- **Status** (stav procesu)

Dále je potřeba nalézt vztahy mezi těmito entitami v jakých společně vystupují.

V tomto případě jsou to tyto relace:

- MA_PROJEKTY (Workspace, Projekt)
- PROCES_V_PROJEKTU (Proces, Projekt, Status)
- UZIVATEL_V_ROLI_VE_WORKSPACE (Uživatel, Role, Workspace)
- UZIVATEL_V_ROLI_V_PROJEKTU (Uživatel, Role, Projekt)
- RODIC_PROCESU (Proces, Proces) - hierarchie procesů
- PREDCHUDCE_PROCESU (Proces, Proces) - zajištění sekvence procesů

Entity konkrétních procesů

Pro úplnost zde popíši základní princip zapojení procesů do jádra systému. Kvůli naprosté různorodosti jednotlivých procesů je potřeba mapovat téměř 1:1 proces na typ entity. Tzn., co proces, to tabulka v databázi. Atributy procesů většinou také 1:1 odpovídají vstupním polím formulářů procesů.

Některé procesy však bývají poněkud komplikovanější z hlediska nároků na strukturu uložených dat a je potřeba většího počtu databázových tabulek, což ale nepůsobí velký problém. Je třeba jen zvolit jednu tabulku, která bude mít vazby na ostatní tabulky nebo bude alespoň schopna se ke všem datům dostat skrze cizí klíče.

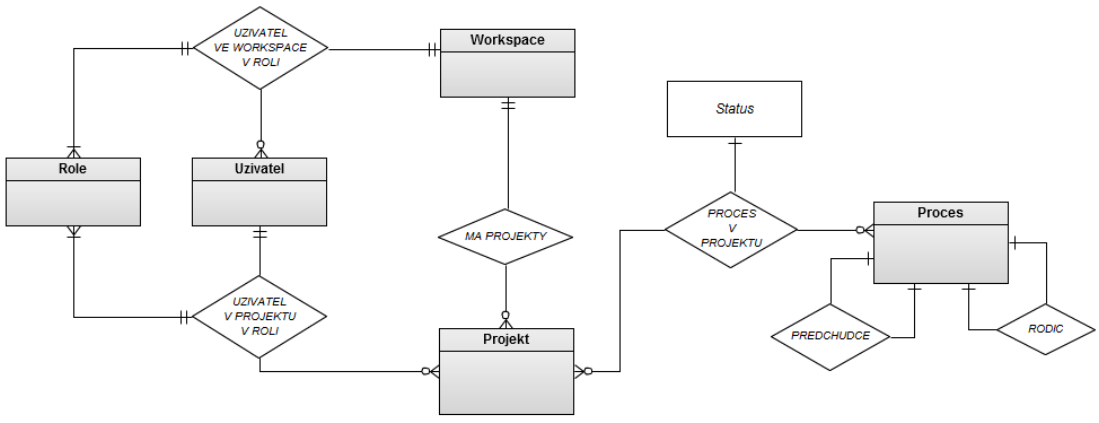
4.1.1 Modelování databázového schématu

Vytváření schématu databáze je při budování informačního systému činností, se kterou se musel setkat snad každý vývojář. Jde o velice zodpovědnou úlohu, kterou si je vhodné předem velmi dobře rozmyslet. Nedostatky v takovém návrhu totiž mohou odsoudit budoucí aplikaci k záhubě. Nesprávným návrhem se můžeme dostat do situace, že požadována data budou obtížně získatelná anebo naprosto znemožní samotné budoucí úpravy.

Pro vytváření modelu struktury systému se v současnosti nejčastěji využívají sémantické nebo objektové datové modely.

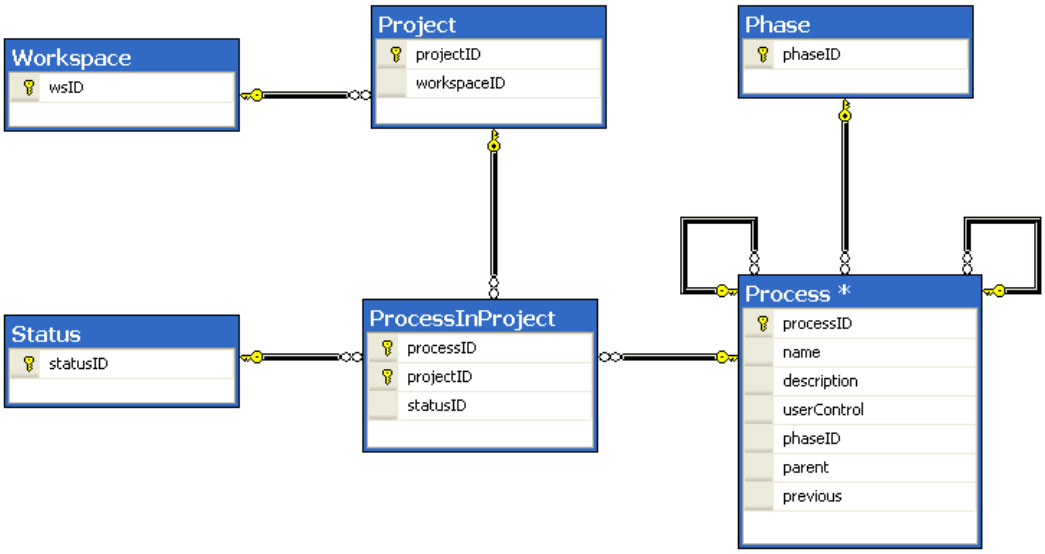
Nejznámějším sémantickým modelem je Entitně-Relační, tzv. ER model. Je to způsob grafické reprezentace logických vazeb mezi entitami za účelem návrhu databáze. Entity jsou abstrakcí určité oblasti reality a jsou jednoznačně určeny svými atributy.

Já jsem se rozhodnul pro modelování použít ER diagram. Zpracování funkčního jádra systému v ER diagramu na konceptuální úrovni je na následujícím obrázku Obrázek 4-1:



Obrázek 4-1 ER digram jádra IS

ER diagram (**Obrázek 4-1**) znázorňuje konceptuální úroveň modelu, která však není databázově zpracovatelná díky vazbám M:N, proto je nyní nutné tyto vazby rozložit. Návrh databáze, tentokrát už formou databázového diagramu je na obrázku **Obrázek 4-2**.



Obrázek 4-2 DB diagram jádra IS

Řešený problém

Z popisu navržené metodiky vyplývá, že procesy jsou řazeny do definované posloupnosti, tzn., že procesy mají své následovníky a předchůdce. Mohou se také hierarchizovat, jinými slovy mohou mít své potomky/rodiče. Všechny procesy mají svůj název, popis a další atributy společné pro všechny procesy. Jak ale navázat atributy jednotlivých „konkrétních“ procesů, když každý z nich bude mít naprosto odlišné atributy? Některé rodičovské procesy zase naopak ani žádné atributy nemají, jde o procesy pouze seskupující své potomky.

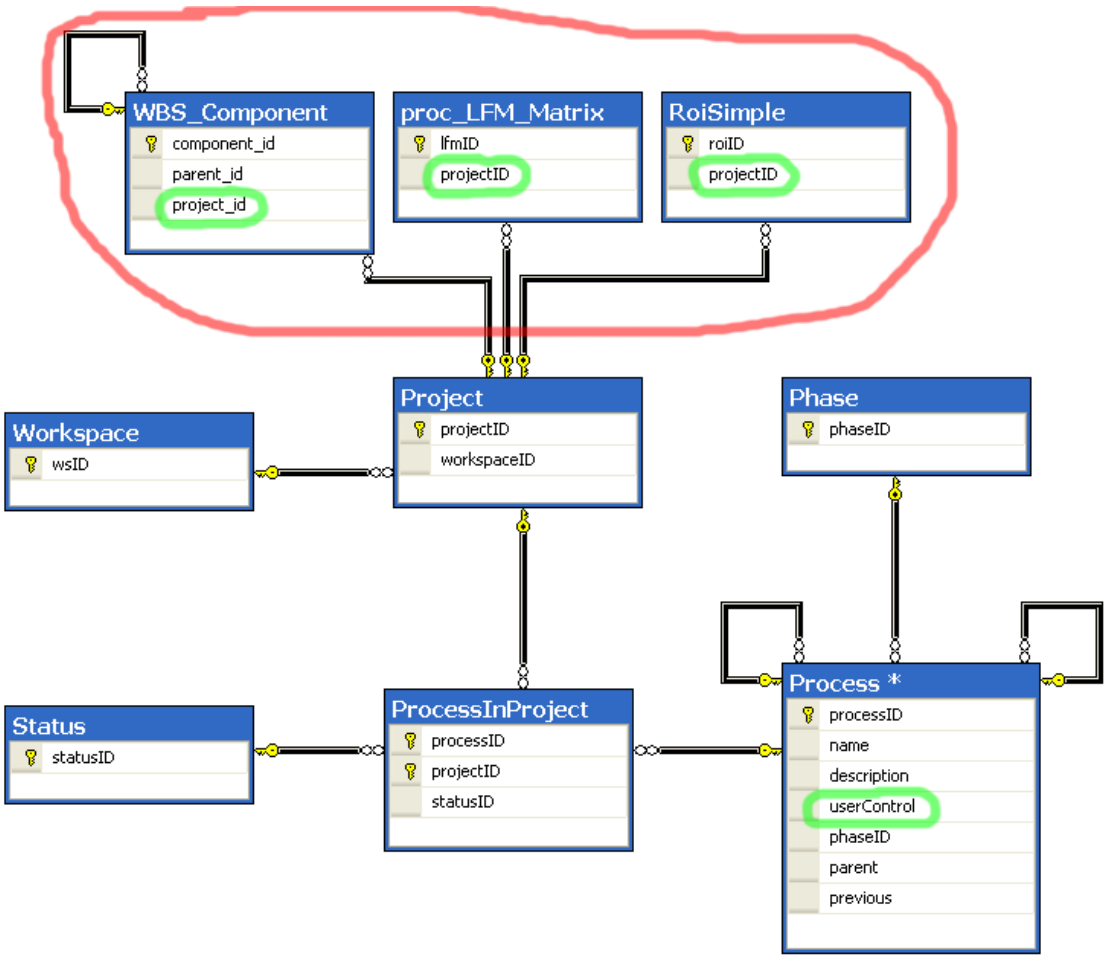
Řešení

Vytvoříme tabulku pro obecný proces, zde přidáme cizí klíč označující předcházející proces jako referenci na sebe sama, který je NULL v případě, že předcházející proces není (tzn., jedná se o první proces v sekvenci) a totéž ještě pro vytvoření hierarchie přidáním cizího klíče rodič. Tím vyřešíme návaznost a strukturu procesů vyžadovanou navrženou metodologií.

Stále však zbývá vyřešit co s konkrétními procesy. Nabízí se možnost vytvoření samostatné tabulky pro každý konkrétní proces (tzn., tabulky LFM, WBS, Rizika, atd.). Jak a kam ale tyto tabulky navázat? Když je naváženeme na tabulku obecného procesu, kam přidat cizí klíč? Když ho přidáme do tabulky konkrétního procesu a bude ukazovat na obecný proces, jak pak v aplikaci při požadavku na získání dat procesu s daným ID jako parametrem zjistíme, kterou tabulku

máme k obecné připojit? Co třeba přidat do tabulky obecného procesu atribut s názvem tabulky konkrétního procesu? Pravděpodobně řešitelné, ale ukázalo se jako nevhodné.

Tímto problémem jsem se zabýval velmi hluboce, jelikož jeho řešení je naprosto klíčové pro způsob řešení celého systému a zpracováváni procesů. Zvolil jsem řešení, které dělá celý IS zcela modulárním. Strukturu výsledné databáze a způsob navázání tabulek procesů zobrazuje diagram (ObrázekObrázek 4-3).



Obrázek 4-3 DB diagram řešení modularity procesů

Toto řešení rozděluje systém z pohledu databáze na dva základní podsystémy.

Jádro systému

Pro jádro systému prozatím poslouží dvě základní tabulky evidující pracovní prostory a projekty v nich. Ve finále toto jádro doplňuji o evidenci typů zdrojů, které jsou k dispozici pro daný pracovní prostor a také kalendář jejich využití.

Otevřenost aplikované metodologie

Celý popis navržené metodologie zde lze vyřešit pouze jedním objektem proces, kterému odpovídá jedna tabulka v DB se dvěma na sebe sama odkazujícími vazbami. Jedna pro určení posloupnosti procesů v metodice odkazující na ID procesu, který mu předchází. Je-li proces první v řadě a nemá tedy žádný proces, který by mu předcházel, je tato hodnota prázdná. Druhá vazba slouží pro seskupování procesů či jejich hierarchizaci. Je to vazba odkazující na rodiče procesu. Tímto způsobem lze vytvořit celý strom procesů. A díky sekvenční vazbě dokonce celou sekvenci sekvenčních stromů.

Z tohoto řešení vyplývá naprostá nezávislost na jedné konkrétní metodice. Umožňuje v systému aplikovat dokonce více metodik najednou přičemž každý projekt může používat jinou.

Je třeba zmínit vyznačený atribut tabulky *Process* s názvem *userControl*. Jde o název komponenty procesu, která implementuje daný proces konkrétně. Více se této komponentě budu věnovat později v kapitole o návrhu procesů (kapitola 4.3.).

Modul procesy

Data konkrétních procesů budeme uchovávat v tabulkách navázaných na tabulku *Project*. Každá může mít svou vlastní strukturu a pracovat s ní bude jen a pouze komponenta procesu, na jejíž název odkazuje výše zmíněný atribut *userControl* tabulky *Process*.

Při přidání dalšího procesu do metodologie stačí z pohledu databáze přidat pouze záznam do tabulky *Process* s názvem a dalšími informacemi o procesu a přidat pro něj do databáze novou tabulku pro potřebnou evidenci dat tohoto procesu. Nová tabulka jen musí obsahovat cizí klíč odkazující na projekt, kterému daný záznam patří.

Konkrétní procesy (typy entit) mohou tvořit celé subsystémy, tzn., mohou se skládat hned z několika typů entit. V takovém případě je zapotřebí navrhnout strukturu databáze komponenty tak, aby zde vystupovala jedna tabulka reprezentující celou komponentu provázaná s ostatními tabulkami, obsahující povinný atribut ID projektu, jakožto cizí klíč odkazující na konkrétní projekt.

Tímto způsobem lze v systému aplikovat i celou další metodologii řízení projektů. Tabulka *Process* později přibírá další atribut identifikující metodologii, které náleží. Pro práci se systémem to následně znamená, že postačí zvolit požadovanou metodologii, tím se vyfiltrují pouze procesy této metodologie a vnitřní vazby procesů pak určí její strukturu (posloupnost a zanoření).

Pro usnadnění další představy o celém systému je vhodné si uvést návrh jeho architektury.

4.2 Návrh architektury systému

Pro zajištění maximální nezávislosti a obecnosti systému a usnadnění jeho testování je vhodné použít rozložení do čtyř základních vrstev.

4.2.1 Prezentační vrstva

První vrstva je vrstva prezentační, která zahrnuje *ASPX* stránky a jejich kód na pozadí a slouží jakožto vrstva zobrazující data uživateli. Zároveň zachytává jeho vstupy a interakce se systémem. Tato vrstva tvoří grafické uživatelské rozhraní.

4.2.2 Vrstva byznys logiky

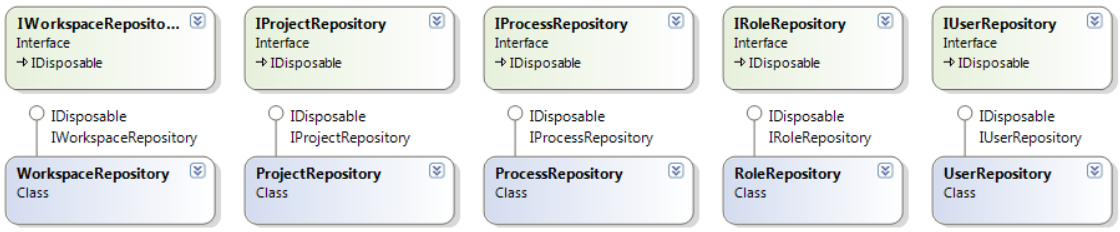
Tato vrstva je obecně používána ve vývoji aplikací pro zajištění její byznys logiky. Ani zde nemá jinou funkci. Je to vrstva mezi prezentační vrstvou a databází. Jediným rozdílem v tomto konkrétním případě a díky rozložení celkově na vrstvy čtyři je, že tato vrstva nevolá přímo základní databázové operace, tzv. CRUD, ale metody nižší vrstvy, která tyto operace zastupuje. Podrobněji toto popíši dále.

4.2.3 Repozitář - vrstva pro přístup k datům

Tato vrstva obsahuje třídy repozitáře, které obsahují metody pro základní CRUD operace pro přístup k datům. Existence této vrstvy přináší výhodu nezávislosti aplikace na konkrétním modelu. Tzn., že např. při změně databáze (např. přechod z MSSQL na Oracle) jen vyměníme repozitář za jiný a zbytek aplikace zůstává beze změn. Metody byznys logiky volají pořád ty samé metody repozitáře i celá byznys logika v nich zůstává beze změn.

Další výhodou zavedení repozitáře je možnost testování. V tomto případě jen zaměníme běžný repozitář za testovací, který nemusí pracovat s databází, ale třeba s testovacími kolekcemi dat přímo v testovacím projektu.

Repozitář je složen ze třídy repozitáře a jeho rozhraní. Pro přehlednost je vhodné vytvořit repozitář pro každý typ entity zvlášť, tak jako na následujícím obrázku Obrázek 4-4.



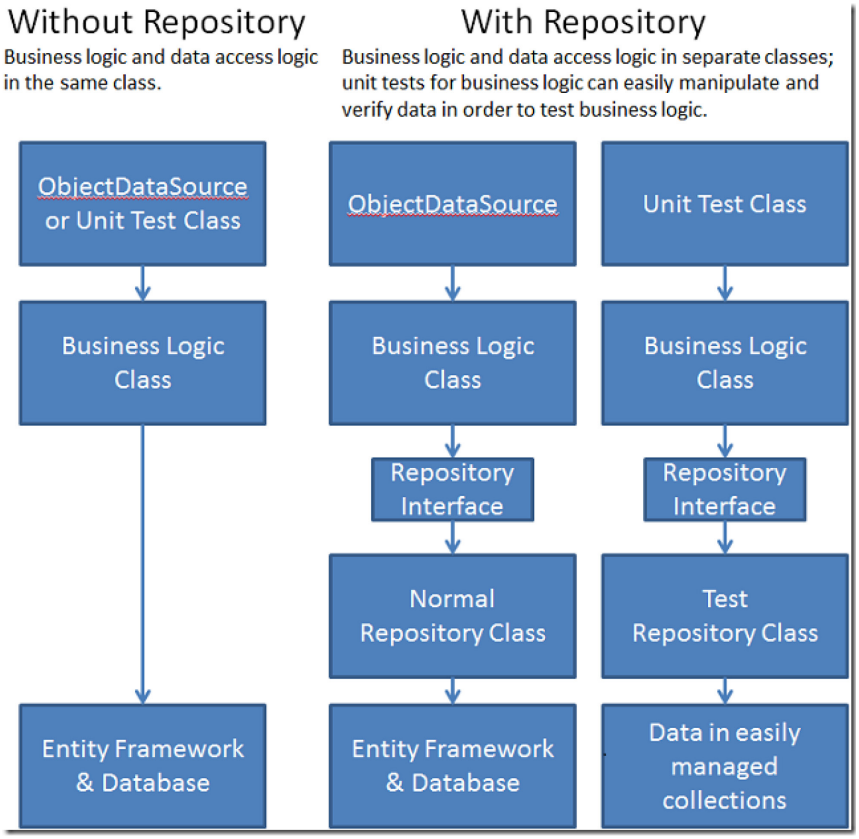
Obrázek 4-4 Třídní diagram: vrstva pro přístup k datům

4.2.4 Model

Model reprezentuje datové úložiště. Pro tuto aplikaci jsou data uložena v databázi MSSQL běžící na SQL Serveru 2008 R2. Struktura této databáze vychází z diagramů databáze obsažených v této kapitole. Jelikož kompletní diagram je příliš rozsáhlý, nemá smysl ho zde uvádět. Jako náhrada může posloužit diagram tříd na obrázku Obrázek 4-7, jelikož v tomto případě jsou tabulky databáze namapované na třídy 1:1. Pro vytvoření tříd (modelu)

z databázových tabulek bude použit Entity Framework 4.0, který tento proces maximálně usnadňuje a poskytuje přívětivé uživatelské prostředí ve Visual Studiu. Entity Framework umožňuje několik přístupu pro vytvoření modelu. Tato aplikace používá Entity Framework 4.0 Database First. Jedná se o přístup, kdy chceme vytvořit model z již existující databáze. Jelikož databáze byla navržena jako první, nabízí se tento přístup jako nejvhodnější.

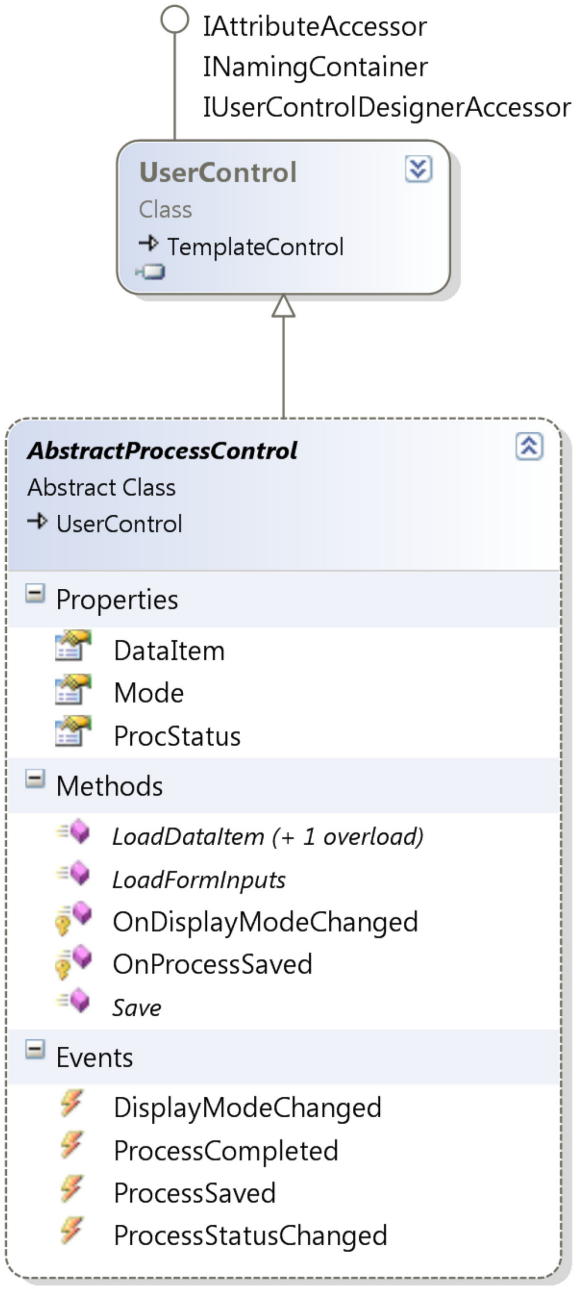
Architektura této aplikace (s repositářem) je zachycena na následujícím obrázku Obrázek 4-5 ve srovnání s běžnou třívrstvou architekturou bez repositáře.



Obrázek 4-5 Srovnání rozvrstvení aplikace bez repositáře a s repositářem **(10)**

4.3 Návrh komponent procesů

Jednotlivé procesy jsou tedy řešeny jako komponenty, konkrétně v ASP.NET pomocí ovládacích prvků UserControls, což je typ komponenty, kterou lze vytvářet ve VisualStudiu i vizuálně přetahováním ovládacích prvků na stránku pomocí drag&drop. Toto přináší téměř neomezené možnosti pro vytvářenou komponentu/ovládací prvek. Pro tyto komponenty jsem navrhnul abstraktní třídu AbstractUserControl, kterou musí daná komponenta implementovat. Obecné operace s procesem už zajišťuje abstraktní třída sama. Struktura této třídy je vyobrazena na obrázku Obrázek 4-6.



Obrázek 4-6 Abstraktní třída pro UserControls implementujících jednotlivé procesy

4.3.1 Struktura abstraktní třídy komponent

Vlastnosti komponenty

Uvedl jsem zde tři základní vlastnosti. Vlastnost *DataItem* je typu *object* a reprezentuje data konkrétního procesu, který pouze přetypuje třídu obsahující svá data na obecný *object*.

Vlastnost *Mode* slouží pro určení módu zobrazení komponenty. Jedná se o typ *enum* mající dvě položky: *READ_ONLY, EDIT*. Jejím smyslem je umožnění zobrazit proces buďto jako formulář pro editaci nebo jen jako výpis pouze pro čtení.

ProcStatus, je vlastnost označující stav procesu.

Metody

Zde bych zmínil tři nejdůležitější metody, které musí třída konkrétního procesu implementovat.

- *LoadDataItem(ID projektu)* – tato metoda očekává parametr obsahující identifikátor projektu pro který chceme načíst data procesu. Komponenta, která implementuje tuto abstraktní metodu si pak vyhledá záznam ze své tabulky (tabulka konkrétního procesu, např. *WBS_Component*) odpovídající požadovanému projektu a načte jej do objektu *DataItem*
- *LoadFormInputs()* – metoda naplní vstupní pole formuláře daty z *DataItem*
- *Save()* – metoda pro uložení dat ze vstupů do databáze

Ukázku implementace jedné konkrétní komponenty uvedu později v kapitole implementace.

4.4 Nakládání s komponentami

V následující části se budu věnovat návrhu jak nakládat s navrženými komponentami v průběhu práce se systémem, tzn. v průběhu řízení projektu.

Řízení projektu probíhá formou plnění jednotlivých procesů. Vždy tedy pracujeme s nějakou komponentou procesu. Jelikož jsou tyto komponenty navrženy jako ovládací prvky *UserControl* a veškerá náplň činnosti procesu je v nich obsažena, postačí jen jedna *aspx* stránka pro zobrazování a práci s nimi. Tuto stránku budu dále nazývat jako řídící stránku procesů.

4.4.1 Řídící stránka procesů

Tato aplikace bude využívat ASP.NET stránky s kódem na pozadí. To znamená, že každá *aspx* stránka sloužící pro zobrazování má k sobě příslušný soubor s funkčním kódem, v tomto případě v jazyce C#.

ASPX stránka

V této stránce je specifikováno pouze rozložení stránky (layout). Jelikož je hlavní stránkou určující layout rozhraní celého systému ještě další nadřazená stránka tzv. *MasterPage*, která obsahuje navigaci a veškeré společné prvky pro celý systém, zbývá zde vyřešit jen umístění načítané komponenty a tlačítek pro interakci s ní. Pro tento účel postačí panel jakožto kontejner

pro zobrazení komponenty a k tomuto vytvořit lištu tlačítek, která budou po kliknutí vyvolávat příslušné události v komponentě (např. tlačítko uložit).

Kód řídící stránky

Kód na pozadí této stránky je zodpovědný za načítání a komunikaci s komponentami procesů. Jeho činnost zde ve velmi zjednodušené formě popíši.

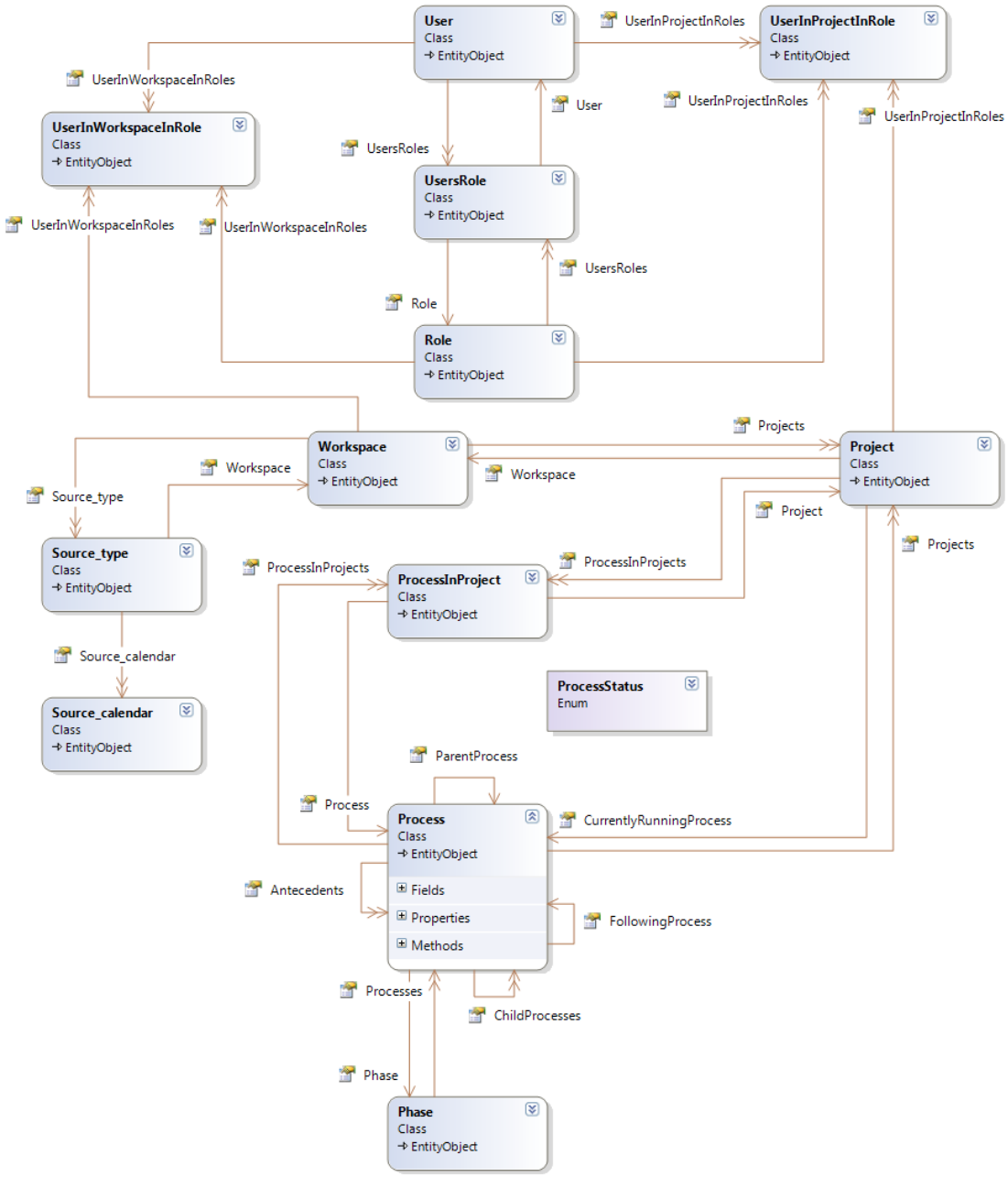
1. Stránce je v parametru URL předán identifikátor procesu, který chceme načíst.
2. Ta se pokusí nalézt proces v databázi v tabulce *Process* a z tohoto záznamu načte název komponenty z atributu *userControlName*.
3. Z adresáře komponent se pak pokusí načíst komponentu s daným názvem přetypovanou na abstraktní *AbstractUserControl*.
4. V případě úspěchu pak zavolá metodu *LoadDataItem*, která naplní konkrétními daty vlastnost *DataItem*.
5. Následně zavolá metodu *LoadFormInputs*, pro vyplnění vstupů z *DataItem*.
6. Nakonec se stránka vykreslí na obrazovku včetně komponenty a načtených vstupů.

Konkrétní ukázku implementace uvedu později v kapitole o implementaci.

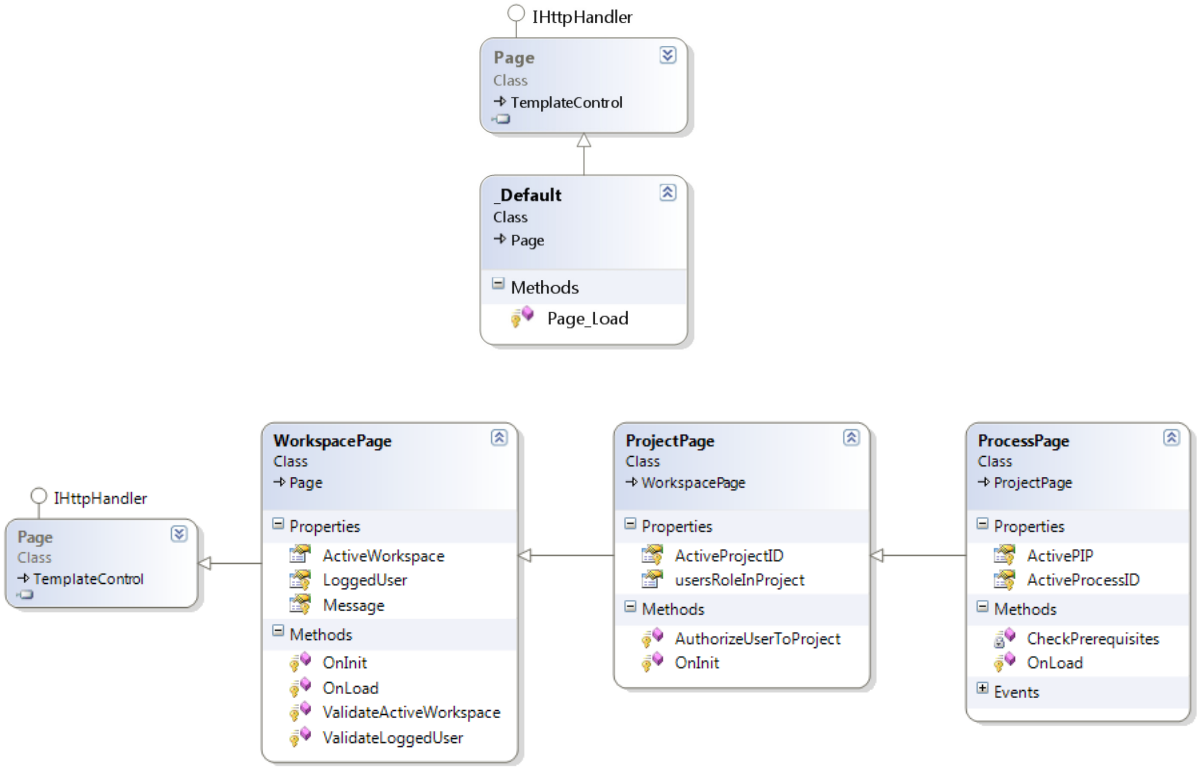
4.5 Dílčí shrnutí

Základní požadavky kladené na IS jsou dány navrženou metodikou řízení projektu. Tato specifikuje vývoj projektu jako deterministickou posloupnost procesů (tj. činnosti vykonávané při řízení projektu) rozdělenou do několika fází. Z toho vyplývá, že v systému je zapotřebí vést evidenci projektů, jednotlivých fází a procesů v nich. Procesy, jakožto typ entity jsou velmi různorodé, proto je vhodné zavést tento typ entity jako jakýsi referenční typ všech procesů přičemž každý jeden konkrétní proces je samostatný typ entity. Referenční entity obsahují pouze základní údaje o procesu: název, popis, atp., plus název komponenty konkrétního procesu, která se načte do stránky. Tyto komponenty jsou řešeny v ASP.NET pomocí ovládacích prvků UserControls, což je typ komponenty, kterou lze vytvářet ve VisualStudiu i vizuálně přetahováním komponent na stránku pomocí drag&drop. Takto vytvořená komponenta implementující funkcionalitu procesu pak musí povinně dědit z abstraktní třídy AbstractUserControl, jejíž strukturu znázorňuje obrázek Obrázek 4-6.

Celý dosavadní návrh doplněný o správu uživatelů a rolí a dalších doposud opomíjených prvků shrnuji v diagramu tříd na obrázku Obrázek 4-7 **Třídní diagram funkčního jádra IS**.



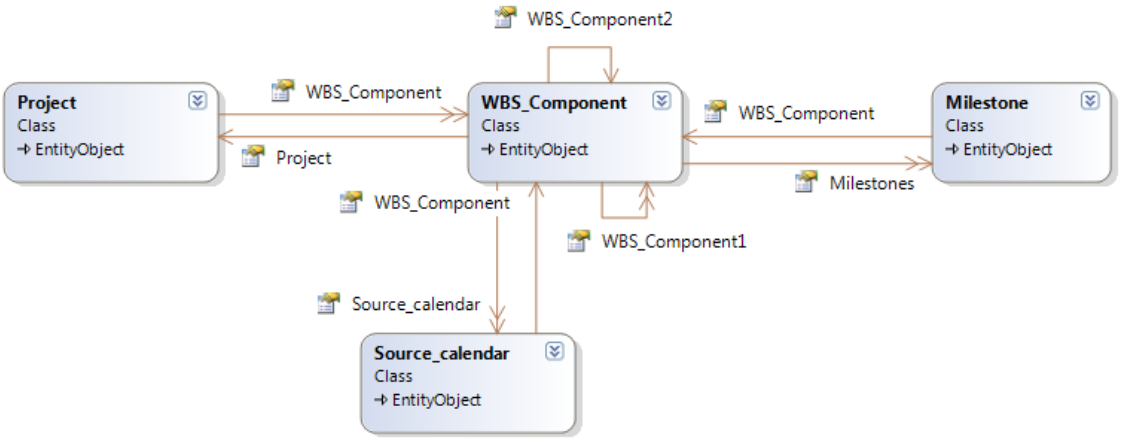
Obrázek 4-7 Třídní diagram funkčního jádra IS



Obrázek 4-8 Úrovně bezpečnosti děděním z třídy System.Web.UI.Page

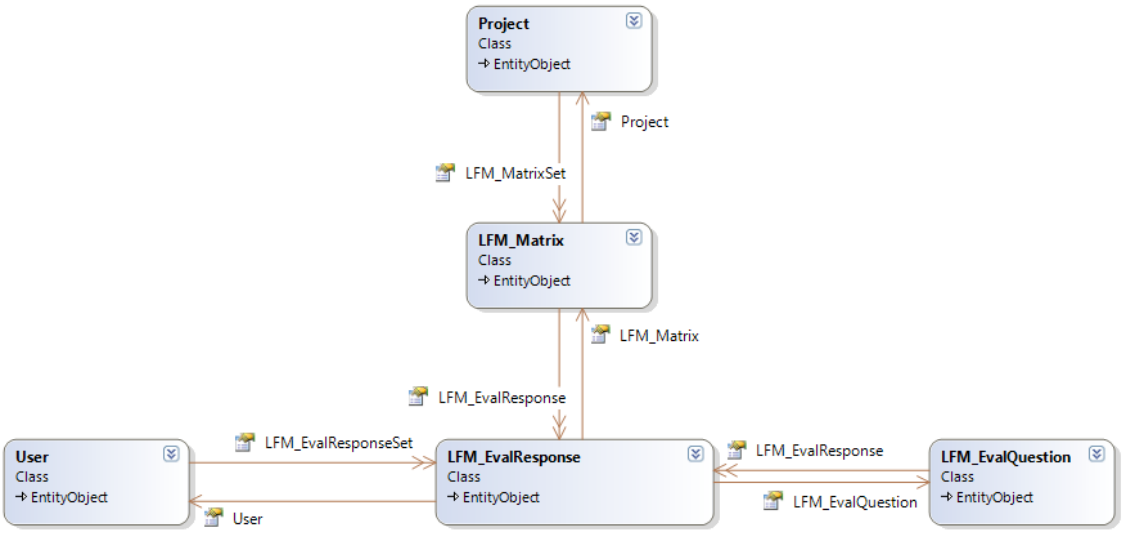
4.6 Ukázky třídních diagramů jednotlivých procesů

4.6.1 WBS



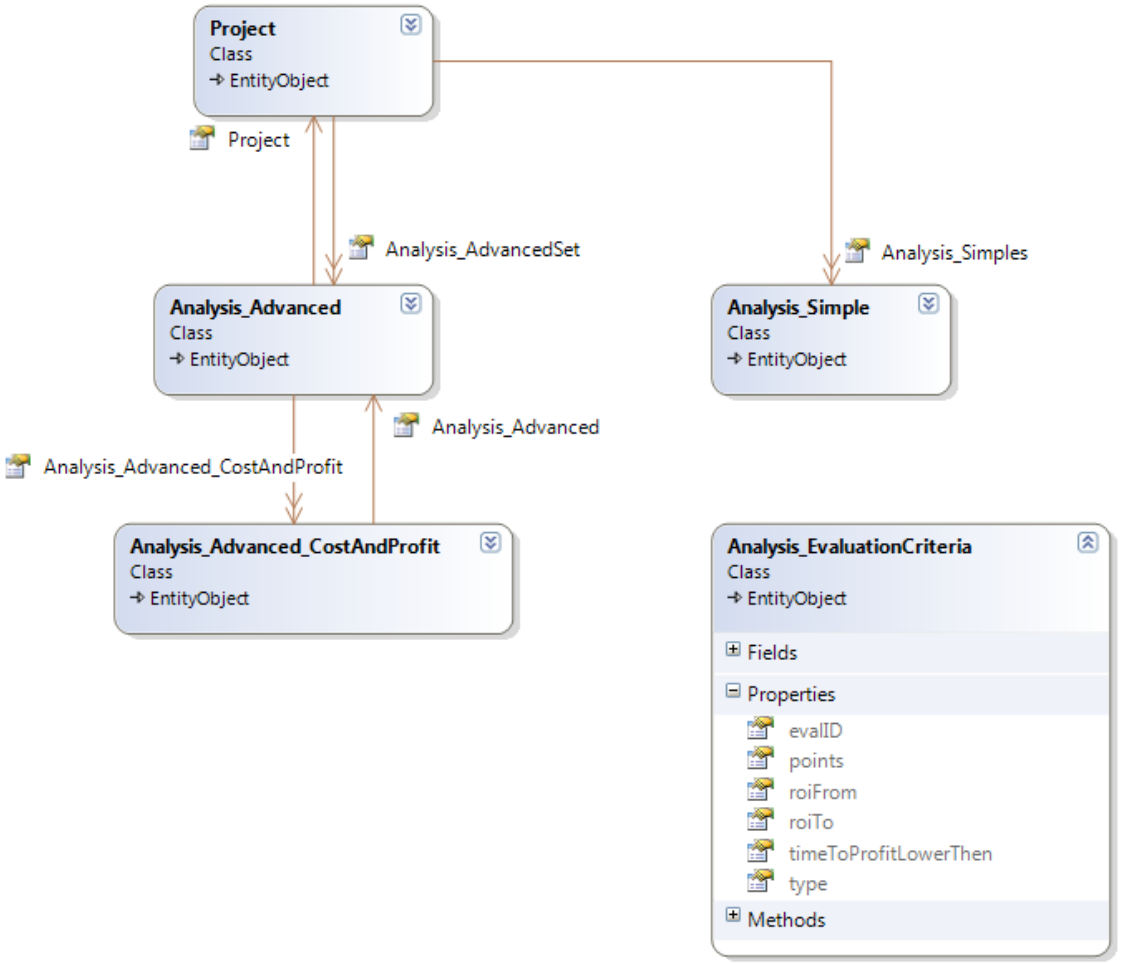
Obrázek 4-9 Třídní diagram procesu WBS

4.6.2 LFM



Obrázek 4-10 Třídní diagram procesu LFM

4.6.3 Analýzy



Obrázek 4-11 Třídní diagram procesu analýzy

5 Implementace

Systém je naimplementován pomocí technologie ASP.NET při použití jazyka C# a systému řízení báze dat MS SQL. Jako vývojové prostředí bylo použito MS Visual Studio 2010. Jako nástroj k objektově-relačnímu mapování (ORM) jsem použil Entity Framework 4.

5.1 Struktura aplikace

Celé řešení systému (solution) je tvořeno jednou web-site a čtyřmi knihovnami tříd.

- Web-site (aspx stránky, prezentační vrstva)
- Knihovna BusinessLogicLayer (vrstva byznys logiky)
- Knihovna DataAccessLayer (vrstva pro přístup k datům)
- Knihovna MyControls (obsahuje vlastní CustomControls a CompositeControls)
- Knihovna Test (unit testy)

Toto rozvrstvení aplikace poskytuje nezávislost aplikační logiky na modelu díky přidání abstrakční vrstvy tříd repositáře mezi byznys vrstvu a model.

5.2 Ukázky implementace

5.2.1 Řídící stránka procesů

Níže uvádím ukázku implementace metody *OnLoad*, řídící stránky procesů, která se spouští při každém načtení stránky (i postback). Podrobný popis této stránky je popsán v kapitole 4.4.1.

```
protected void Page_Load(object sender, EventArgs e)
  {
        if (ActivePIP != null)
        // Checks existencion of ProcessInProject
        {
            ltrProcessName.Text = ActivePIP.Process.name;
            string controlName = ActivePIP.Process.path;
            string ControlVirtualPath = String.Format("~/UC/{0}.ascx", controlName);
            try
            // try to load user control
            {
                processControl = LoadControl(ControlVirtualPath) as AbstractProcessControl;
            }
            catch (Exception ex)
            {
                Session["Message"] = ex.Message;
                Response.Redirect(Server.MapPath("~/Process/"));
                //throw ex;
            }
            if (!IsPostBack)
            //on first page load initialize data for user control and save them to viewstate
            {
                processControl.LoadDataItem(ActivePIP);
                ViewState["pcDataItem"] = processControl.DataItem;
                processControl.Mode = DisplayMode.VIEW;
            }
            else
            {
                processControl.DataItem = ViewState["pcDataItem"];
            }
            // fill input form fields with data from DataItem
            processControl.LoadFormInputs();
            processControl.ProcessSaved += new EventHandler(ProcessControlSaved);
            // place user control on page
            ControlPlaceHolder.Controls.Add(processControl);

            // Process action bar with action buttons (save,skip,export,etc.)
            ActionBar.SaveDraftRequest += new EventHandler(ActionBar_SaveDraftRequest);
            ActionBar.SaveCompleteRequest += new EventHandler(ActionBar_SaveCompleteReq);
            ActionBar.SkipRequest += new EventHandler(ActionBar_SkipRequest);
        }
    }
```

Výpis 5-1 Metoda OnLoad řídící stránky procesů

Závěr

Hlavním cílem mé diplomové práce byl návrh a následná implementace IS pro správu a řízení projektů dle metodiky navržené kolegy Jakubem Štolfou (1) a Ondřejem Koběrským (2).

V teoretické části shrnuji základní východiska, nezbytná pro vypracování této práce, vyjasňuji základní pojmy z oblasti projektového řízení a popisuji použité nástroje a programovací jazyky.

V praktické části nejdříve specifikuji základní požadavky na implementaci IS, tyto požadavky analyzuji a na jejich výsledcích provádím vlastní návrh a realizaci implementace IS.

U mnou předloženého návrhu řešení bych vyzdvihnul modularitu výsledného IS a jeho nezávislost na aplikované metodice řízení projektů. Díky tomuto je IS v budoucnu připraven na další rozšiřování a úpravy, případně aplikovat i více metodik.

Jako pozitivní rovněž vidím, že již od počátku byla diplomová práce směřována na požadavky dvou reálných společností, se kterými jsme měli možnost v průběhu vývoje komunikovat a získávali tak velmi cenné poznatky z praxe. Vzhledem k tomu, že tyto společnosti o navrhované řešení projevily potencionální zájem, má výstup z diplomové práce reálné vyhlídky na uplatnění v praxi a případně i další pokračování.

Mým osobním přínosem a velmi zajímavou zkušeností bylo kromě zpracování atraktivního tématu odzkoušení si týmové spolupráce, hledání způsobu komunikace a předávání informací, stanovování dílčích cílů a hodnocení aktuálního stavu práce.

Literatura

1. **Štolfa, Jakub.** *Metodologie řízení projektů za pomocí agilních přístupů.* Ostrava : VŠB-TU Ostrava, 2011. Diplomová práce.
2. **Koběrský, Ondřej.** *Metodologie řízení projektů za pomocí agilních přístupů.* Ostrava : VŠB-TU Ostrava, 2011. Diplomová práce.
3. **Martínek, Z., Kreslíková, J.** Management projektů. *Vysoké učení technické v Brně, Fakulta informačních technologií.* [Online] prosinec 2007. https://www.fit.vutbr.cz/study/courses/MPR/private/mpr-opora-v2.pdf.
4. **Fiala, Petr.** Projektové řízení – manažerská strategie projektově orientovaných firem. *www.odbornecasopisy.cz.* [Online] 12. 4 2011. [Citace: 12. 4 2011.] http://www.odbornecasopisy.cz/index.php?id_document=29030.
5. **Institute, Project management.** *Guide to the Project Management Body of Knowledge.* Newtown Square : Project Management Institute, 2004. ISBN 1-930699-45-X.
6. **Commerce, Office of Government.** The definitive PRINCE2 project management training resource. *PRINCE2.com.* [Online] ILX Group, UK . http://www.prince2.com.
7. **ING. ZUZANA ŠOCHOVÁ, MBA.** AGILNÍ METODY A ŘÍZENÍ PROJEKTŮ. *JAVA portál.* [Online] [Citace: 22. červenec 2011.] www.java.cz/dwn/1003/27929_Abstract-AgilniMetody_RizeniProjektu.pdf.
8. **Martin Fowler, Jim Highsmith.** FHT-Stuttgart related materials. *Andrey's place - Hristov.com.* [Online] srpen 2001. [Citace: 22. červenec 2011.] http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf.
9. **Běhálek, Marek.** Programovací jazyk C#. [Online] 2007. [Citace: 15. červenec 2011.] http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/index.html.